Technologie Sieciowe Lista 2

Piotr Szymański

Treść zadania

Rozważmy model sieci S = <G,H>. Przez N=[n(i,j)] będziemy oznaczać macierz natężeń strumienia pakietów, gdzie element n(i,j) jest liczbą pakietów przesyłanych (wprowadzanych do sieci) w ciągu sekundy od źródła v(i) do ujścia v(j).

- Zaproponuj topologię grafu G ale tak aby żaden wierzchołek nie był izolowany oraz aby: |V|=20, |E|<30. Zaproponuj N oraz następujące funkcje krawędzi ze zbioru H: funkcję przepustowości 'c' (rozumianą jako maksymalną liczbę bitów, którą można wprowadzić do kanału komunikacyjnego w ciągu sekundy), oraz funkcję przepływu 'a' (rozumianą jako faktyczną liczbę pakietów, które wprowadza się do kanału komunikacyjnego w ciągu sekundy). Pamiętaj aby funkcja przepływu realizowała macierz N oraz aby dla każdego kanału 'e' zachodziło: c(e) > a(e).
- Niech miarą niezawodności sieci jest prawdopodobieństwo tego, że w dowolnym przedziale czasowym, nierozspójniona sieć zachowuje T < T_max, gdzie: T = 1/G * SUM_e(a(e)/(c(e)/m - a(e))), jest średnim opóźnieniem pakietu w sieci, SUM_e oznacza sumowanie po wszystkich krawędziach 'e' ze zbioru E, 'G' jest sumą wszystkich elementów macierzy natężeń, a 'm' jest średnią wielkością pakietu w bitach. Napisz program szacujący niezawodność takiej sieci przyjmując, że prawdopodobieństwo nie uszkodzenia każdej krawędzi w dowolnym interwale jest równe 'p'. Uwaga: 'N', 'p', 'T_max' oraz topologia wyjściowa sieci są parametrami.
- Przy ustalonej strukturze topologicznej sieci i dobranych przepustowościach stopniowo zwiększaj wartości w macierzy natężeń. Jak będzie zmieniać się niezawodność zdefiniowana tak jak punkcie poprzednim (Pr[T < T_max]).
- Przy ustalonej macierzy natężeń i strukturze topologicznej stopniowo zwiększaj przepustowości. Jak będzie zmieniać się niezawodność zdefiniowana tak jak punkcie poprzednim (Pr[T < T_max]).
- Przy ustalonej macierzy natężeń i pewnej początkowej strukturze topologicznej, stopniowo zmieniaj topologię poprzez dodawanie nowych krawędzi o przepustowościach będących wartościami średnimi dla sieci początkowej. Jak będzie zmieniać się niezawodność zdefiniowana tak jak punkcie poprzednim (Pr[T < T_max]).

Opis rozwiązania

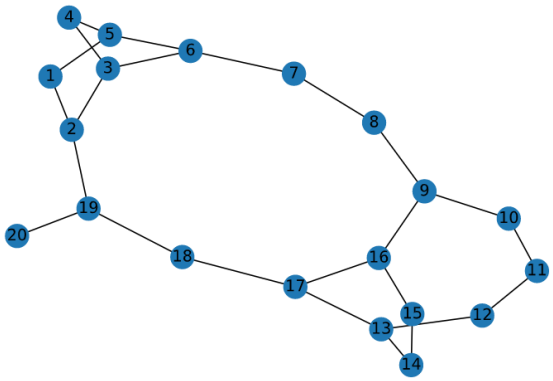
Zadanie zostało rozwiązane przy pomocy języka Python. Do operowania na grafach używam biblioteki *networkx*, a do rysowania wykresów biblioteki *matplotlib.* Dodatkowo w celu operacji na liczbach używam biblioteki *random* oraz *numpy.*

```
import networkx as nx
import matplotlib.pyplot as plt
import random
import numpy as np
```

1. Na początku tworzę graf o 20 wierzchołkach i 24 krawędziach, poniższym kodem.

```
def gen_graph():
    graph = nx.Graph()
    graph.add_nodes_from([i for i in range(1,21)])
    for i in range(1,20):
        graph.add_edge(i,i+1)
    # dodanie dodatkowych 5 losowych krawędzi
    for _ in range(5):
        i = random.randint(1,20)
        k = i
        while k == i:
            k = random.randint(1,20)
        graph.add_edge(i,k)
    nx.draw(graph, with_labels=True)
    plt.show()
    N = gen_intensity_matrix(graph, 50, 200)
    return graph,N
```

Topologia grafu jest następująca:



Następnie generuję macierz N natężeń strumienia pakietów. Do tego posłużyła mi poniższa funkcja:

```
def gen_intensity_matrix(graph, low, high):
    """
    argumenty low i high są odpowiednio dolnym i górnym ograniczeniem
    przedziału, z którego losuję wartości natężeń
    """
    size = graph.number_of_nodes()
    N = np.zeros((size,size))
    for i in range(size):
        for j in range(size):
            if i == j:
                N[i][j] = 0
            else:
                N[i][j] = random.randint(low,high)
    return N
```

Wartość macierzy N jest następująca:

[[0. 107. 66. 137. 120. 58. 72. 53. 103. 101. 98. 139. 166. 141. 165. 169. 123. 158. 81. 64.]
[128. 0. 192. 169. 93. 50. 114. 93. 192. 165. 145. 143. 155. 138. 151. 186. 107. 199. 197. 68.]
[63. 67. 0. 87. 109. 59. 90. 65. 189. 84. 112. 159. 72. 91. 108. 107. 104. 153. 92. 73.]
[100. 155. 70. 0. 74. 93. 126. 182. 53. 90. 64. 85. 68. 169. 156. 128. 159. 108. 139. 180.]
[174. 110. 63. 116. 0. 180. 77. 125. 139. 200. 74. 180. 101. 91. 112. 176. 88. 189. 56. 167.]
[98. 72. 141. 84. 146. 0. 111. 160. 184. 133. 74. 146. 129. 199. 90. 98. 92. 92. 186. 162.]
[136. 175. 182. 189. 158. 173. 0. 133. 185. 127. 77. 184. 187. 51. 80. 111. 194. 161. 137. 69.]
[83. 118. 127. 132. 167. 154. 175. 0. 167. 122. 136. 136. 124. 65. 155. 72. 98. 137. 73. 118.]
[175. 95. 151. 61. 96. 180. 77. 62. 0. 145. 105. 134. 178. 196. 76. 65. 195. 115. 195. 69.]
[182. 141. 189. 195. 149. 136. 137. 171. 113. 0. 93. 165. 54. 112. 96. 139. 90. 187. 60. 153.]
[195. 80. 132. 181. 196. 136. 126. 131. 126. 165. 0. 149. 94. 132. 127. 180. 199. 75. 77. 124.]
[149. 73. 135. 187. 63. 163. 134. 114. 54. 95. 74. 0. 80. 108. 106. 166. 150. 78. 95. 195.]
[135. 131. 128. 90. 69. 137. 194. 144. 100. 70. 177. 178. 0. 195. 90. 172. 139. 180. 116. 65.]
[173. 56. 199. 152. 97. 163. 120. 143. 60. 110. 147. 151. 58. 0. 148. 180. 52. 137. 67. 181.]
[93. 76. 156. 60. 160. 185. 86. 141. 101. 126. 103. 176. 59. 156. 0. 182. 142. 176. 127. 167.]
[82. 50. 162. 63. 130. 135. 198. 56. 74. 189. 191. 87. 63. 124. 165. 0. 66. 130. 188. 128.]
[163. 86. 72. 112. 162. 108. 119. 172. 98. 121. 57. 159. 76. 150. 132. 76. 0. 109. 99. 72.]
[83. 132. 76. 135. 167. 185. 99. 96. 59. 133. 121. 53. 138. 163. 104. 163. 137. 0. 181. 200.]
[83. 126. 57. 164. 92. 124. 133. 171. 139. 98. 133. 173. 175. 63. 116. 65. 77. 92. 0. 129.]
[94. 50. 180. 70. 144. 88. 161. 181. 123. 92. 160. 123. 71. 52. 175. 51. 162. 73. 137. 0.]]

Za funkcję przepustowości **c** przyjąłem średnią wartość natężeń tablicy N pomnożoną przez wylosowaną stałą z konkretnego zakresu. Posłużył mi do tego poniższy kod:

```
def set_c_function(graph, intensity_matrix, range):
    """ zmienna range to para liczb ograniczających przedział"""
    size = graph.number_of_nodes()
    avg_intensity = np.sum(intensity_matrix) // (size*size-size)

    for e in graph.edges():
        multiplier = random.randint(range[0], range[1])
        graph[e[0]][e[1]]['c'] = avg_intensity*multiplier
```

Przyjąłem, że pakiety będą przesyłane najkrótszą ścieżką łączącą dwa wiechołki. Wtedy każda krawędź zawarta w takiej ścieżce zostanie obciążona natężeniem tego przesyłu danych. Niech więc funkcja przepływu **a(e)** będzie równa sumie natężeń wszystkich przesyłów danych, które będzie musiała obsłużyć. Poniższy kod przedstawia jej implementację:

```
def set_a_function(graph, intensity_matrix):
    size = graph.number_of_nodes()

    for i, row in enumerate(intensity_matrix,1):
        for j, intensity in enumerate(row,1):
            if i != j:
                path = nx.shortest_path(graph, i, j)
                for p in range(len(path)-1):
                    graph[path[p]][path[p+1]]['a'] += intensity
```

2. Następnym elementem zadania jest oszacowanie niezawodności sieci.

Do wykonania tego zadania napisałem poniższą funkcję:

```
def reliability(N, p, T_max, graph, avg_packet_size, capacity_range):
    """
    p - prawdopodobieństwo nie uszkodzenia krawędzi
    avg_packet_size - średni rozmiar pakietu czyli wartość m
    capacity_range - zakres, z którego będzie wybierany mnożnik do ustalenia przepustowości
    """
    delays = is_working(p, graph, N, avg_packet_size, capacity_range)

    if len(delays) != 0:
        connected_num = len(delays)/1000
        avg_delay = sum(delays)/len(delays)

        good_T_num = 0
        for d in delays:
            if d < T_max:
    else:
        connected_num = 0
        avg_delay = 0
        good_T_num = 0

    return connected_num, avg_delay, (good_T_num/1000)
```

Na początku za pomocą funkcji *is_working* sprawdzam czy graf jest spójny.

```
def is_working(p, graph, N, avg_packet_size, capacity_range):
    nx.set_edge_attributes(graph, p, 'p')
    delays = []
    for _ in range(1000):
        g = nx.Graph(graph)
        for e in g.edges():
            if g.get_edge_data(*e).get('p') < random.random():
                g.remove_edge(*e)

        if not nx.is_connected(g):
            continue

        set_a_function(g, N)
        set_c_function(g, N, capacity_range)

        if not check_capacity(g):
            continue

        delays.append(delay(g, N, avg_packet_size))

    return delays
```

W funkcji tej jest wykonanych 1000 testów sprawdzających czy sieć nie zostanie uszkodzona oraz czy pomimo pewnych uszkodzeń nadal nadaje się do użytku (czyli graf nadal jest spójny, a poziom przepływu nie przekracza poziomu przepustowości).

```
def check_capacity(g):
    for e in g.edges():
        if g.get_edge_data(*e).get('a') > g.get_edge_data(*e).get('c'):
            return False
    return True
```

1Funkcja zwracająca wartość False, gdy a(e) > c(e) dla, którejś z krawędzi oraz True, taka sytuacja nie zachodzi

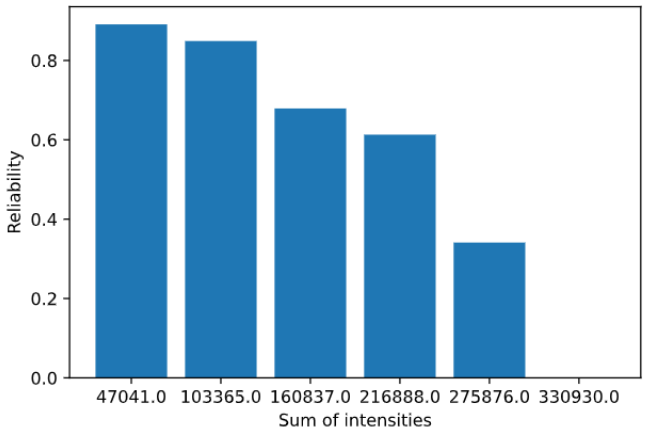
```python
def delay(graph, intensity_matrix, average_packet_size):
    G = np.sum(intensity_matrix)
    m = average_packet_size
    sum_e = 0
    for e in graph.edges():
        a = graph.get_edge_data(*e).get('a')
        c = graph.get_edge_data(*e).get('c')
        sum_e += a/(c/m - a)

    return sum_e/G
```

Funkcja licząca średnie opóźnienie

Dla grafu pokazanego na samym początku, przy parametrach
p = 0,95, T_max = 0.0001, capacity_range = (400,600) i avg_packet_size = 1
wyniki są następujące:

```
Chance of network working properly:  0.886
Average delay:0.000081
Reliability:  0.756
```

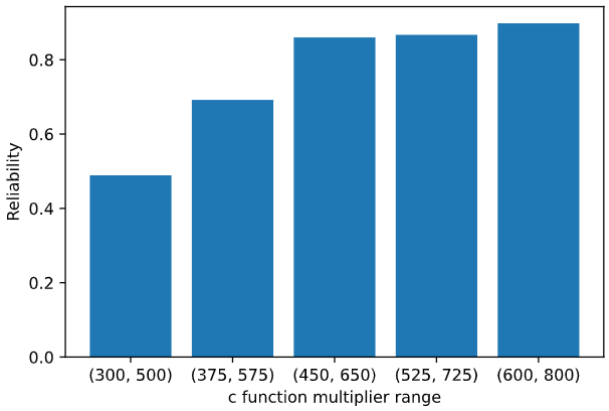
3. Następnym zadaniem było sprawdzenie jak będzie się zmieniać niezawodność sieci przy zwiększaniu wartości w macierzy natężeń N.



Wykres przedstawiający niezawodność grafu od wzrastającej sumy natężeń macierzy N

Badania zostały przeprowadzone dla stałych c = 131072, T_max = 0,00005, p = 0,95 m = 1. Widzimy tutaj jak wzrastająca wartość natężeń co raz skuteczniej '*zapycha*' sieć co powoduje spadek jej niezawodności.

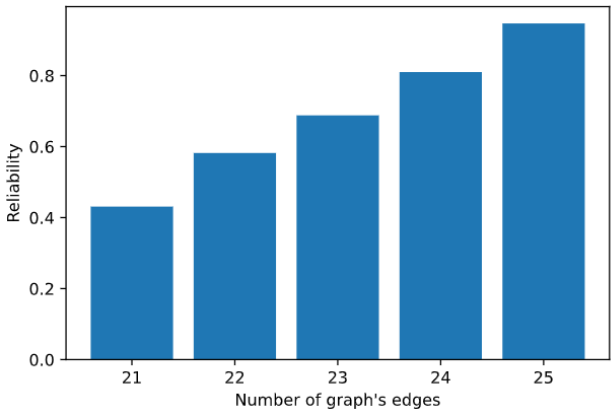
4. Kolejnym zadaniem było sprawdzenie jak będzie się zmieniać niezawodność sieci przy zwiększaniu przepustowości sieci.



Wykres przedstawiający niezawodność grafu od wzrastającej przepustowości

Badania zostały przeprowadzone dla stałych p = 0,95, T_max = 0,0001, m = 1.
Z wykresu możemy wywnioskować, że zwiększająca się przepustowość zwiększa nam również niezawodność sieci. Dzieje się tak dlatego, że dzięki możliwości przesłanie większej ilości danych w jednej sekundzie, maleje średnie opóźnienie.

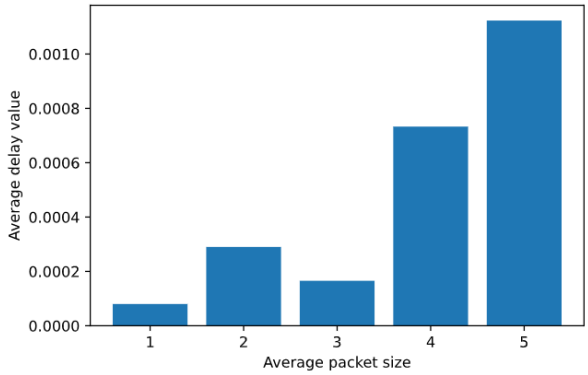
5. W następnym podpunkcie polecenie polega na zbadaniu niezawodności sieci przy jednoczesnym dodawaniu kolejnych krawędzi do grafu.



Wykres przedstawia wartość niezawodności do wzrastającej liczby krawędzi grafu sieci

Badania zostały przeprowadzone dla stałych: p = 0,95, T_max = 0,0001, m = 1
oraz c_range = (400, 600). Wykres bardzo dobrze nam obrazuje jak niewielki przyrost liczby krawędzi powoduje znaczny wzrost niezawodności sieci. Tutaj dodanie 4 krawędzi spowodowało około dwukrotny wzrost wartości niezawodności tej sieci.

6. Wpływ średniej wielkości pakietu na średnie opóźnienie



Wartość średniego opóźnienia do wzrastającej średniej wielkości pakietu

Badania przeprowadzone dla stałych: p = 0,95, T_max = 0,0002, c_range = (400, 600). Wykres bardzo dobrze obrazuje ogromny wpływ średniego rozmiaru pakietu na średnie opóźnienie. Widzimy, że wzrost średniego rozmiaru pakietu z wartości 1b na wartość 5b spowodował dziesięciokrotny przyrost średniego opóźnienia.

7. Wnioski

Przeprowadzone badania oraz napisane programy pozwalają bardziej zrozumieć architekturę sieci oraz jej możliwe słabe punkty. Dobra sieć, odporna na wszelkie uszkodzenia powinna posiadać dużą ilość krawędzi oraz jak najmniej wierzchołków, do których prowadzi jedna ścieżka. Ważne jest też, aby ustalić przepustowość odpowiednią do potrzeb i zastosowania sieci oraz nie dopuścić do zbyt dużego natężenia strumienia pakietów, ponieważ może to znacznie wpłynąć na niezawodność naszej sieci.

Przedstawienie sieci za pomocą grafu bardzo ułatwia analizę, co przyczynia się do uniknięcia wielu istotnych błędów.